RIELAC, Vol.XXXIII 1/2012 p.16-24 Enero - Abril ISSN: 1815-5928

RIELAC
Revista de Ingeniería Electrónica, Automática y Comunicaciones

"Análisis y diseño de software para la sincronización de intersecciones semafóricas"

Ing. Eileen Cardoso Espinosa, Dr. Valery Moreno Vega

Teléfono:2602055

e-mail: eileen@electrica.cujae.edu.cu, *valery@electrica.cujae.edu.cu*

RESUMEN / *ABSTRACT*

El presente trabajo hace uso de las técnicas de programación orientada objeto para el diseño de una aplicación de software con el objetivo de crear un sistema que permita sincronizar intersecciones de una arteria semaforizada. Este sistema permite obtener mediante el uso de herramientas de ingeniería de tránsito un plan de tiempo de luces para las intersecciones y tiempos de desfasamiento entre los semáforos de las intersecciones para lograr la sincronización.
El sistema se desarrolló en el entorno de desarrollo multiplataforma Qt.
Palabras claves: Sincronización de intersecciones, ingeniería de tránsito, RUP.

This paper deals with the analysis and design of a software application to synchronize semaphores situated along an artery of a city. With the software the system´s operators will obtain a timed sequence for lights changes as well as an offset of time between the intersections located in the artery that warranties the desired synchronization. The system was developed using Qt.
Keywords: intersection synchronization, traffic engineering, RUP.

INTRODUCCION

A medida que pasa el tiempo, el número de vehículos en las calles de las ciudades del mundo crece. Este proceso genera congestionamiento y aumento de los accidentes, por lo que para su atenuación, el uso de semáforos ha alcanzado un notable desarrollo, implicando que se hayan tenido que crear sistemas de control de tráfico avanzados que incluyen la coordinación computarizada y la incorporación de detectores de vehículos, que dependiendo de su variación hacen que cambie en forma dinámica y continua el tiempo asignado a cada acceso de las intersecciones[9].

Cuba no está ajena a esta problemática y para solucionarla, el Centro Nacional de Ingeniería de Tránsito de Cuba y el Complejo de Investigaciones Tecnológicas Integradas, junto a la colaboración del Departamento de Automática del ISPJAE, se encuentran desarrollando un sistema inteligente de transporte para el control del flujo vehicular y el diseño total de controladores semafóricos, como aspectos fundamentales.
Como parte del control de flujo vehicular, el sistema inteligente de transporte necesita la incorporación de una herramienta de software que permita establecer el sincronismo o coordinación de las intersecciones semaforizadas de una arteria. Esta herramienta constituye el tema principal de este artículo.
La creación de este software, constituye una gran ayuda al trabajo de los especialistas de ingeniería de tránsito en nuestro país. Ellos en la actualidad, realizan innumerables operaciones de cálculo, manualmente, para obtener el sincronismo lo cual conlleva a que se cometan errores y se obtengan valores de menor precisión a los que se pueden obtener mediante un software. A partir de la utilización de un software para la sincronización de arterias semaforizadas los especialistas contarán con una correcta distribución de los tiempos del semáforo en las condiciones de operación establecidas, y a partir de esta información el sistema autónomamente calcula la sincronización.

Incorporando este software al sistema inteligente se podrá actualizar los parámetros de la sincronización con una determinada sistematicidad que permitirá una mejor planificación de los programas de tiempos de los semáforos, alcanzándose una mayor continuidad del flujo vehicular en nuestras calles.
Debido a la importancia y actualidad del tema, en el diseño del software se utilizaron técnicas de ingeniería de software que permitirán que el mismo, en un futuro pueda ser mejorado, actualizado o integrado a otros sistemas inteligentes de transporte.

ASPECTOS GENERALES

Una arteria no es más que una vía constituida por varias intersecciones. Cuando se quiere sincronizar una arteria se espera lograr una continuidad en el flujo de los vehículos. El sistema propuesto logra el sincronismo siguiendo el criterio de "Onda Verde" [1] que tiene como principio conseguir el máximo de recorrido de los vehículos sin encontrar los semáforos en rojo.
El sistema propuesto, se encarga de capturar y almacenar los datos por independiente de cada intersección que conforme la arteria. Durante el período de procesamiento de datos, el sistema, mediante la programación adecuada de un método para el cálculo de la distribución de tiempos de los semáforos conocido como método Webster[3, 10], y la programación de un algoritmo para la sincronización basado en el comportamiento de los diagrama espacio-tiempo[1, 8], se obtiene la distribución de tiempos de los semáforos y el desfasamiento entre las distintas intersecciones respectivamente. Una vez que se tienen estos resultados, es necesario mediante un planificador semafórico[6], programar los semáforos con estos los valores y de esta manera se podrá dar inicio a los conteos de los tiempos de las luces en el instante adecuado, para lograr así la sincronización real de la arteria.
El software como salida muestra una simulación del sincronismo mediante una gráfica y además genera un reporte que informa datos generales de las intersecciones y su plan de tiempo respectivo.

LEVANTAMIENTO DE REQUISITOS

En el diseño de un software es muy importante conocer los requerimientos tanto funcionales como no funcionales. Estos requisitos son definidos a partir de las necesidades del cliente que va hacer uso de la aplicación. Sin los requisitos es muy difícil crear una aplicación que satisfaga completamente objetivos determinados. A continuación se plantean requisitos funcionales y no funcionales del software[2].

Principales requisitos funcionales:

I. Configurar los parámetros generales de la arteria que implica tanto los parámetros que caracteriza cada una de las intersecciones así como sus fases[3] (combinación de uno o más movimientos que reciben simultáneamente el derecho de paso en una intersección) respectivas y los parámetros relacionados con la sincronización de las intersecciones.

II. Deben validarse los parámetros de entrada del sistema.

III. El sistema debe crear un proyecto donde se guarde toda la configuración del sistema. Los especialistas podrán abrir los proyectos y modificarlo según sea necesario.

IV. El sistema debe generar un gráfico (Diagrama Espacio-Tiempo) que no es más que una representación a escala en el sentido horizontal de una arteria, con sus intersecciones. En las calles transversales se colocan columnas en el sentido vertical donde se representa el tiempo necesario para una secuencia completa de todas las indicaciones de señal del semáforo con sus divisiones. Simulando el avance de un vehículo en línea recta a lo largo de la arteria, pueden trazarse líneas diagonales de acuerdo con el tiempo que requieren esos recorridos. Estas líneas, naturalmente, pasarán por la sección correspondiente a la fase verde[1]. Así puede obtenerse una faja o banda dentro de los límites que permita el rojo. Ver figura 1.

V. El usuario debe tener las opciones de imprimir y salvar los reportes que genere el sistema.

VI. El usuario debe tener la posibilidad de variar la velocidad de sincronización del sistema.

VII. El usuario debe tener la posibilidad de especificar y variar la distancia a la que se encuentra una intersección respecto a una de referencia.

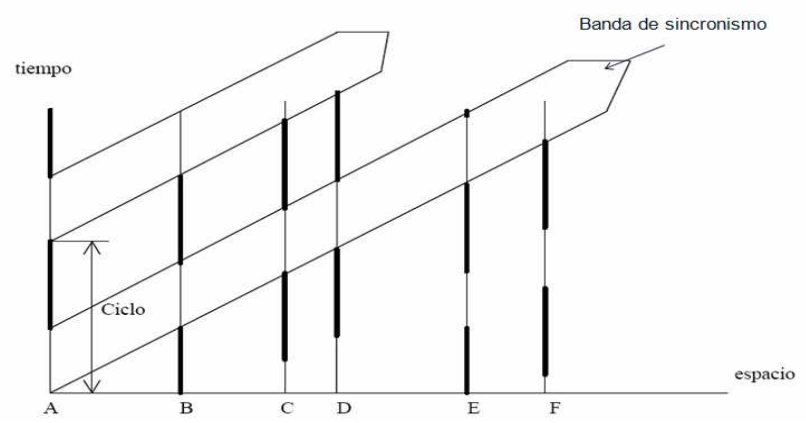

Figura 1

Diagrama Espacio-Tiempo.

Principales requisitos no funcionales:

I. Uso de idioma español en todo el proyecto.

II. Debe hacerse evidente al usuario el uso del sistema internacional de unidades, en la interfaz gráfica del sistema.

III. Uso de colores que se relacionen con las luces de los semáforos (verde, rojo, ámbar)

IV. Utilizar un formato y estilo uniforme en todas las ventanas de la aplicación.

V. Utilizar símbolos que permitan ayudar al usuario a comprender la utilidad de las funcionalidades y representaciones del sistema.

CASOS DE USO

A partir del levantamiento de requisitos se decide definir los casos de uso de manera que encierren procesos independientes que ocurren dentro del sistema. En la figura 2 se muestra un diagrama de casos de uso[5, 7] que sugiere el comportamiento del Sincronizador Semafórico[2].

Para la ejecución de los casos de uso se requiere de la acción del Usuario o Especialista en Ingeniería de Tránsito, que se encargará de la configuración del sistema de manera que los parámetros introducidos correspondan con las características reales de la arteria semaforizada en la que se esté trabajando.

Figura 2

Casos de Uso asociados al funcionamiento del Sistema.

El caso de uso "Configurar Sistema" es un proceso que tiene como propósito, crear proyecto y configurar todos los parámetros para realizar el sincronismo de diferentes intersecciones de la arteria semaforizada. El proyecto después de ser configurado debe ser guardado en el formato que defina el sistema. Con este caso de uso se da respuesta a los requisitos funcionales (I, II, III, VI, VII) y cumple con los requisitos no funcionales.

"Generar Reporte de Intersecciones" crea un reporte de cada intersección, permitiendo que el usuario pueda tener un resumen detallado de todos los parámetros que necesita así como los resultados de la distribución de tiempos que genera el sistema, partiendo de la configuración introducida hasta ese instante. El sistema da la opción de guardar el reporte. Antes de ejecutar este caso de uso debe ejecutar "Configurar Sistema". Con este caso de uso se da respuesta al requisito funcional (V) y cumple con los requisitos no funcionales.

"Graficar Diagrama de Espacio-Tiempo" con este caso de uso el sistema le permite al usuario analizar a través de la gráfica que representa el Diagrama de Espacio-Tiempo cómo se comportará la sincronización de la arteria en correspondencia con la parametrización que se haya establecido. Partiendo del análisis de este diagrama el especialista puede decidir si la velocidad para la que concibió el sincronismo es la más adecuada. Este caso da respuesta al requisito funcional (IV) y cumple con los requisitos no funcionales.

DISEÑO DE LA ESTRUCTURA DEL SOFTWARE

En esta sección se desarrolla la arquitectura del software, partiendo de un análisis más profundo de los principales casos de uso, para poder diseñar las clases que describan la problemática de este trabajo y poder establecer las relaciones que tienen entre sí. Los flujos de trabajos fundamentales en esta etapa de diseño son:

- El análisis de la arquitectura, de cuyo proceso se obtiene un diagrama de las clases de análisis de impacto en la arquitectura.
- Análisis de casos de uso: donde se obtienen diagramas de clases y diagramas de secuencia que realizan este caso de uso.
- Análisis de las clases: donde se asignan las responsabilidades a las clases según los roles que juegan dentro de los casos de uso[4].

Para conformar el diagrama de clases se utilizó el modelo de diseño de software "Modelo-Vista- Controlador"[11]. El uso de este modelo facilita el diseño y programación del software porque como el mismo nombre lo sugiere está formado por tres grupos o paquetes de clases que tienen funciones diferentes, *clases modelo* que son las que se diseñan con la finalidad de describir los objetos del sistema, las *clases vista* que son las que se utilizan como interfaz gráfica de usuario y *clases controlador* que hace función de intermediario entre las clases de la interfaz gráfica de usuario, GUI, del sistema con las clases que modelan el sistema. Estas clases contienen en lo fundamental la lógica de negocio o de funcionamiento de la aplicación. En la figura 3 se muestra un esquema con la relación que existe entre las clases del sistema según el modelo utilizado.

El paquete de *clases modelo* está constituido por tres clases Tsincronismo, Tinterseccion y Tfase entre las que existe una relación de composición tal como ocurre en la realidad porque una intersección tiene varias fases y el sincronismo de una arteria no se puede hacer si no se analizan cada una de las intersecciones que la conforman.

El paquete *clases vista* tiene cinco clases: las clases GUI, VentanaPrincipal, VentanaProyecto, VentanInterseccion, VentanaSincronización y VentanaFase, las cuatro últimas están asociadas a VentanaPrincipal.

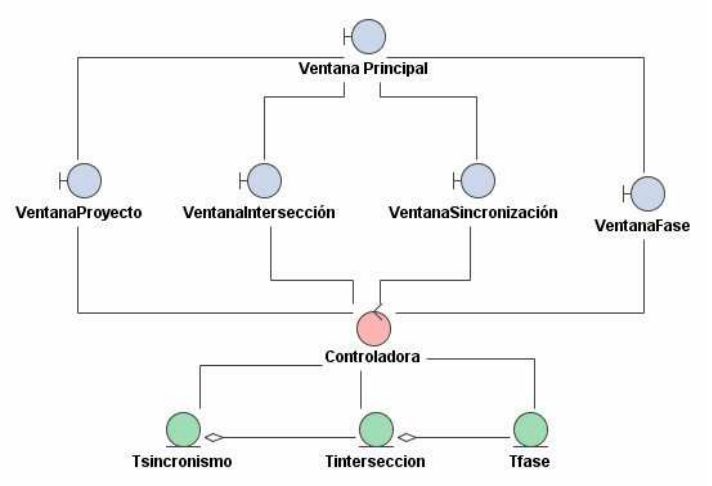

Figura 3

Modelo de diseño de software "Modelo-Vista-Controlador".

Una vez que se tiene el diseño de clases se hace un análisis más profundo de los casos de uso pero desde una perspectiva diferente ya que se analiza la secuencia de interacciones cronológicas entre objetos individuales del sistema para que pueda ejecutarse completamente cada caso de uso.

El caso de uso donde más actividades ocurren es Configuración del Sistema porque abarca desde la creación de un proyecto de sincronismo hasta la edición y configuración de cada elemento del sistema. Por tanto para la ejecución completa de este caso se sigue el orden de acciones:

I. Abrir nuevo proyecto.
II. Editar sincronismo.
III. Agregar intersecciones.
IV. Editar intersecciones.
V. Agregar fases.
VI. Editar fases.
VII. Guardar proyecto.

En este caso de uso el usuario al interactuar con una instancia de la clase VentanaPrincipal genera un proyecto mediante el método (newFile()) al ocurrir esa acción se hace un llamado a la operación mostrar (Mostrar()) de la clase VentanaProyecto de este modo y se abre el nuevo proyecto creado[2].
Asociados también a la clase VentanaPrincipal, existe otros métodos que permiten agregar al proyecto de sincronización, las intersecciones y las fases (interc() y fase ()). Con estos métodos lo primero que se realiza es crear una nueva fase o una nueva intersección en el proyecto en que se esté trabajando y abrir las respectivas ventanas de configuración donde el usuario debe introducir los datos particulares de cada una. Luego se guarda la información en las instancias de las clases del modelo mediante la función GetControlValue.
Cuando el usuario desea editar o modificar las parámetros de algún elemento del sistema, se ejecutan los métodos de edición de la clase VentanaProyecto, estos métodos de manera general buscan una referencia a lo que se quiere modificar, se toman los parámetros almacenados en las instancias de las clases del modelo (SetControlValue) y se muestran en pantalla para que el usuario pueda realizar cambios sobre lo que ya estaba creado para luego guardar los cambios utilizando la misma referencia.
Todo este proceso de configuración del sistema se puede comprender mejor con el diagrama de secuencia[5,7] que aparece en la figura 4.
En el análisis del caso de uso Graficar Diagrama de Espacio-Tiempo, hay que tener en cuenta que para que este pueda ser ejecutado el sistema debe haber sido configurado con antelación. Una vez hecho esto el usuario interactúa directamente con una instancia de la clase VentanaProyecto que tiene un método para graficar la sincronización. Este método se denomina Graficarsincronizacion y tiene como objetivo representar el Diagrama Espacio-Tiempo, gestionando a través del método

Getsincronismo() de la clase controladora los parámetros que necesita para hacer la representación gráfica, luego crea un objeto de tipo RenderArea que es el que contiene todos los métodos necesarios para representar el diagrama. Por último se usa el método show () y se obtiene como resultado la gráfica dentro de una ventana deslizable. En la figura 5 se observa del diagrama de secuencia de este caso de uso.

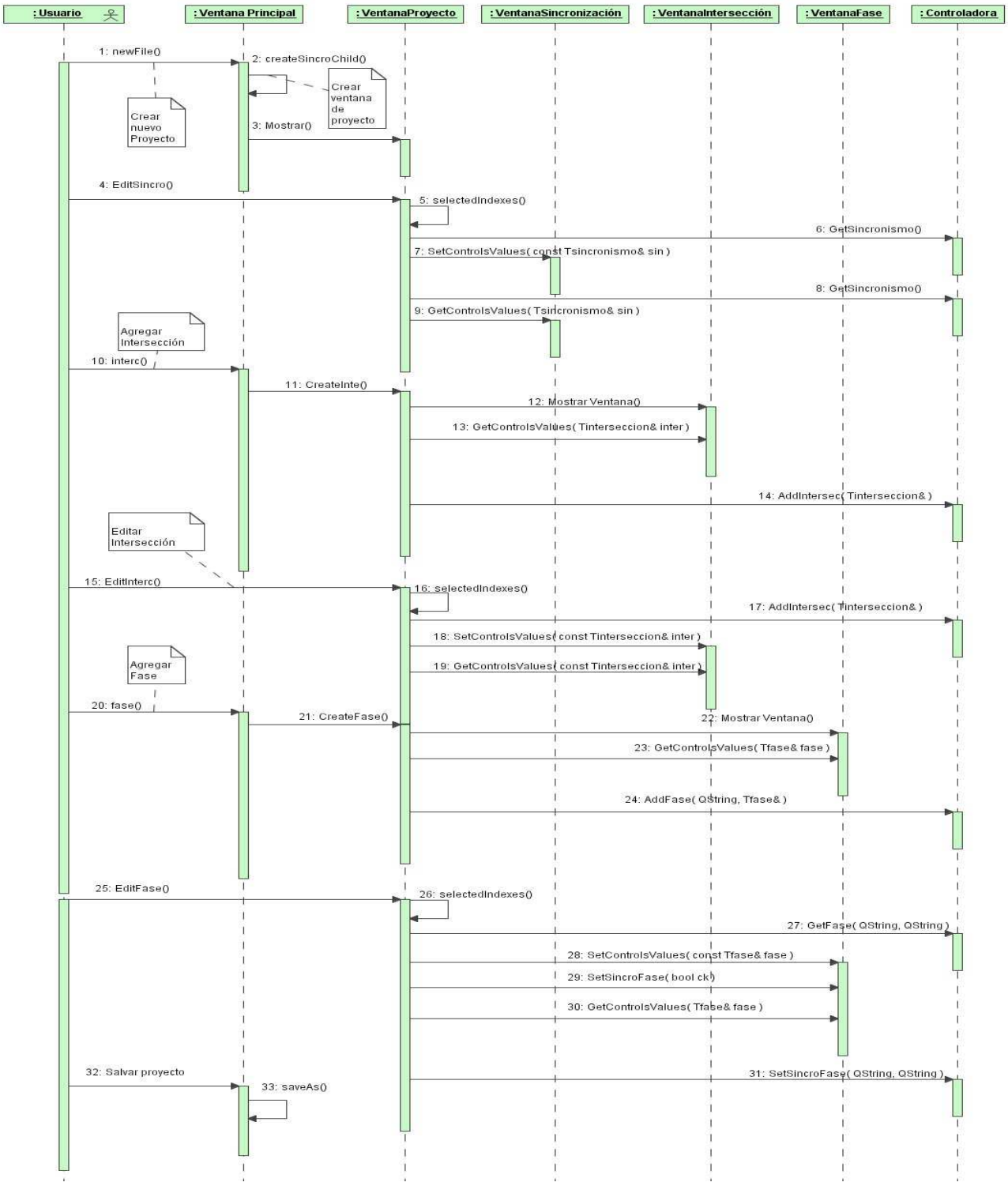

Figura 4
Diagrama de secuencia del caso de uso Configurar Sistema.

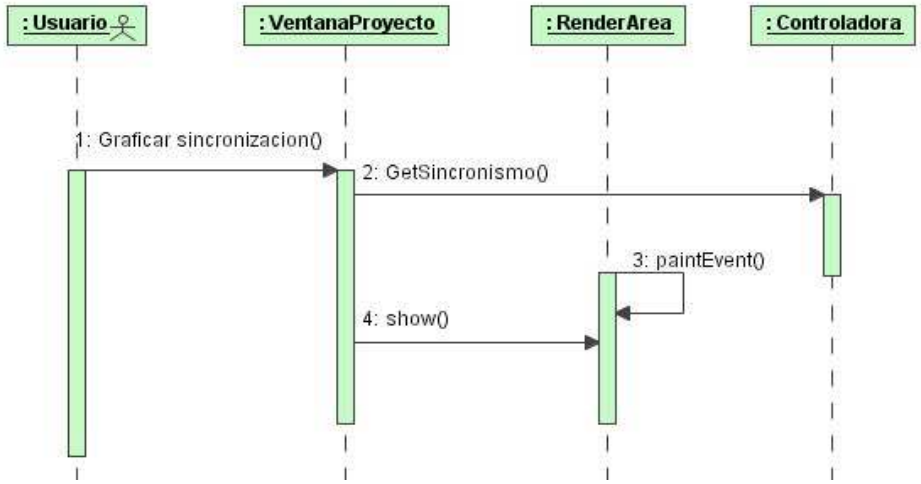

Figura 5
Diagrama de secuencia del caso de uso Graficar Diagrama de Espacio-Tiempo.

Por último queda el análisis de Generar Reporte que al igual que el caso de uso anterior, debe ser ejecutado luego de haberse configurado el que configura el sistema.
Este caso de uso se desencadena cuando se activa el método de la clase VentanaProyecto, Reporte() que invoca GetInterseccion() de la clase Controladora el cual retorna una referencia a la intersección a la que se le quiere hacer el reporte para a partir de ella llegar a los datos de la intersección. Luego teniendo los parámetros necesarios se crea una instancia de la clase VentanaReporte para automáticamente llamar al método CreateReport() que es el encargado de diseñar el formato del reporte. Por último se muestra el reporte mediante show()[2]. Esta secuencia de actividades se puede ver en la figura 6.

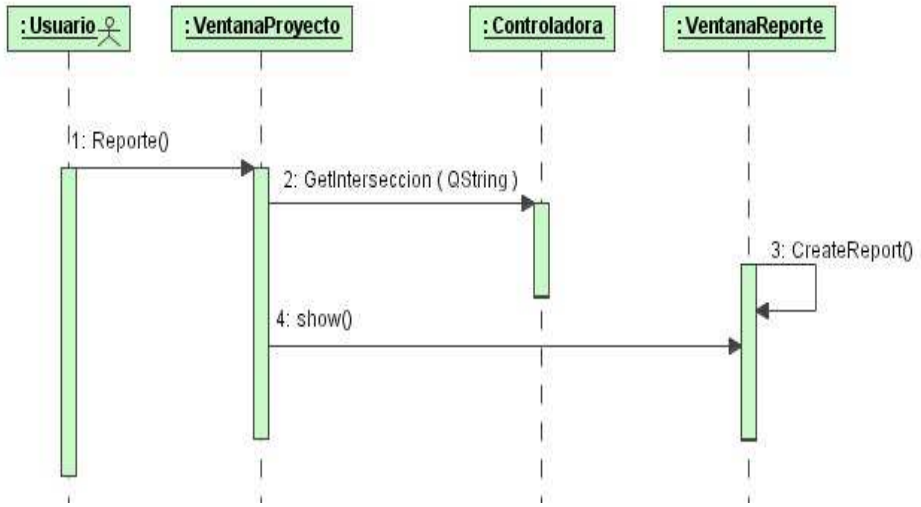

Figura 6
Diagrama de secuencia del caso de uso Generar Reporte de intersecciones.

CONCLUSIONES

En función de un levantamiento de requisitos realizado con el criterio de los expertos en ingeniería de tránsito, se logró definir concretamente los requerimientos tanto funcionales como no funcionales básicos, de un software para sincronizar una arteria semaforizada.
Se utilizó el modelo de diseño de software Modelo-Vista-Controlador pues mediante él se puede crear una relación de clases bastante sencilla que da solución a la problemática de este trabajo.
Usando el Lenguaje Unificado de Modelado (UML) se establecieron los casos de uso de este software y luego de un análisis más detallado se establecieron los diagramas secuenciales[5] de cada uno de los casos de uso que refleja cómo interactúan las instancias de las clases definidas para desarrollar los casos de uso.
Todo este procedimiento permitió la captura de la estructura y comportamiento del sistema para la posterior implementación del mismo.

REFERENCIAS

1. **DÍAZ IVORRA, MARIA DEL CARMEN:** "Métodos geométricos de coordinación de intersecciones reguladas por semáforos", XIV Congreso Internacional de Ingeniería Gráfica, Santander, España, 2002.
2. **CARDOSO ESPINOSA, EILEEN:** "Desarrollo de software para la sincronización de intersecciones semafóricas", ISPJAE, La Habana, Cuba, 2011.
3. **CAL Y MAYOR REYES SPÍNDOLA, RAFAEL:** *Ingenieria de Trásito. Fundamentos y Aplicaciones*,7a.Edición, MéxicoD.F., México, 1994.
4. **FERNÁNDEZ PRIETO, ADEL:** "Análisis y diseño del núcleo de un sistema SCADA con módulo para la detección de fallas", ISPJAE, La Habana, 2005.
5. **RUMBAUGH, JAMES. Jacobson, Ivar. Booch, Grady:** *El Lenguaje Unificado de Modelado. Manual de Referencia*, AddisonWesley,1999.
6. **TRAVIESO VELÁZQUEZ, JUAN JOSÉ. ORTEGA CORTEGUERA, YARAN MANUEL: "**Programación de un Planificador Semafórico". ISPJAE, La Habana, 2010.
7. **SCHULLER, JOSEPH: "**Aprendiendo UML en 24 horas", Prentice Hall, Naucalpan de Juárez, Edo. De México, 2001.
8. **CAL Y MAYOR, RAFAEL. ASOCIADOS: "**Manual de Planeación y Diseño para la administración del tránsito y el trasporte". Bogotá D.C., 2002
9. **GÓMEZ RESTREPO, ALEJANDRO. "**El estado del arte en la modelación de problemas de tránsito". Universidad Nacional de Colombia, 2005.
10. **GONZALEZ CALLEROS, JUAN MANUEL: "**Un modelo heurístico nuevo para el análisis del flujo vehicular". INAOE, Tonantzintla, Agosto 2003**.**
11. **KAISLER, STEPHEN H.:** *Software Paradigms*. Wiley-Interscience, New Jersey.2005.

AUTORES

Eileen Cardoso Espinosa, Ingeniera en Automática, ISPJAE, La Habana, Cuba, eileen@electrica.cujae.edu.cu.

Valery Moreno Vega, Ingeniero en Máquinas Computadoras, máster en Informática Aplicada, Doctor en Ciencias Técnicas, profesor titular, Instituto Superior Politécnico José Antonio Echeverría, 266-3343 valery@electrica.cujae.edu.cu. Actualmente realiza investigaciones en el área de robótica, informática aplicada a la automatización y en aplicaciones para el control utilizando métodos de inteligencia artificial.

Recibido: Noviembre 2011
Aprobado: Febrero 2012